APPLICATION REPORT FROM CONTROL DATA CORPORATION

A METHOD OF GENERATING UNIFORM RANDOM NUMBERS

By W. J. Westlake

CONTROL DATA
CORPORATION

A METHOD OF GENERATING UNIFORM RANDOM NUMBERS

By W. J. Westlake

1. INTRODUCTION

The most widely-used uniform random number generators for digital computers are the so-called congruential generators. A series of numbers x_i is generated recursively by a relation of the type

$$x_i = kx_{i-1} + c \quad \text{modulo } a^n$$

where k, c, a, n and the starting value x_o are integers. The x_i are then divided by a^n to give a sequence of numbers each member of which lies between 0 and 1. If c is zero, the generator is referred to as 'multiplicative' while if c is not zero the generator is referred to as 'mixed'. The numbers produced by congruential generators behave, to a large extent, as if they were a sequence of random samples from a uniform distribution from 0 to 1 (referred to from now on as U(0, 1)) and are widely used in programs where such samples are needed. An excellent summary of properties of these generators can be found in the paper by Hull and Dobell [1].

2. PROPERTIES OF CONGRUENTIAL GENERATORS

These may be conveniently summarized into advantages and disadvantages. The advantages are those of the fixed long-cycle guarantee and the ease of implementation on a digital computer. The numbers x_0, x_1, x_2 ... produced by a congruential generator will all be different until a number x_N is reached such that $x_0 = x_N$. Thereafter the generator will produce the same sequence of numbers and is said to have a cycle of length N. For a mixed congruential generator it is easy to ensure that the cycle length is a^n - the maximum possible - which is, in general, a very large number. While this is not possible for a multiplicative generator, it is still possible to ensure a cycle length almost as large as a^n. The long-cycle guarantee is of particular importance in computer programs which use a large quantity of random material. Programs which simulate the behavior of systems in which random phenomena

occur are an obvious example. Ease of implementation is achieved (for a binary machine, say) by choosing a equal to 2 and the multiplier to be such that multiplication can be performed by one or two left shifts followed by additions.

The general conditions for generators to have maximum cycle length will not be given here (see [1] for complete details). For congruences modulo 2^n they are as follows.

For the mixed generator

$$x_i = kx_{i-1} + c \qquad \text{mod } 2^n,$$

c must be odd and k must be congruent to 1 mod 4 to ensure the maximum cycle length of 2^n.

For the multiplicative generator

$$x_i = kx_{i-1} \qquad \text{mod } 2^n,$$

k must be congruent to ± 3 mod 8 to ensure the maximum cycle length of 2^{n-2}.

The disadvantage of this type of generator is that of any non-random device which is used to generate supposedly random numbers. At best they can be termed pseudo-random numbers, meaning numbers which are produced by a non-random process but which behave to a certain extent as if they were a random sample from the population U(0, 1). However, no final assurance is ever possible that some serious non-random feature will not turn up. Consequently, satisfactory behavior of the generator can be established (and then only tentatively) only on the basis of its passing a number of statistical tests. The literature abounds with testimonials to the excellent statistical properties of one congruential generator or another; but equally abundant are reported examples of such generators having unsatisfactory statistical properties. The starting point for the present investigation is the paper of MacLaren and Marsaglia [2]. They claim that, in most cases reported, the statistical

tests applied to the generated pseudo-random numbers are insufficiently stringent to afford a reasonable test of their randomness. They then set up a fairly stringent series of tests which among other things will test for correlation between the random numbers over spans as large as 10. All the multiplicative and mixed congruential generators on which they used these tests displayed some significantly non-random features. Their solution to the problem lies in proposing two new schemes which, incidentally, both give satisfactory results on the statistical tests. One scheme involves use of buffered storage and the writing into the computer of a large block of random numbers from a satisfactory existing table of random numbers (e.g., Rand Corporation table of 10^6 random digits). The second scheme uses the numbers generated by a mixed congruential generator to shuffle the numbers produced by a multiplicative congruential generator; these shuffled numbers then form the basis for the random numbers. A disadvantage of this latter scheme is that it requires a block of 128 numbers from the multiplicative generator to be stored in the computer at all times during the generation of random numbers. This had led us to study other methods of combining congruential generators which avoid this disadvantage but preserve the desirable long cycle. One such method is now examined.

3. A GENERATOR BASED ON THE COMBINATION OF TWO CONGRUENTIAL GENERATORS

The generator was implemented and tested on the CONTROL DATA ®6600. Since this computer has a register length of 60 bits, arithmetic will be modulo 2^{59} in this instance although the same principles could be employed on a computer of any register length. A succession of pairs of 59-bit words is generated by two distinct congruence relations:

$$x_i = k_1 x_{i-1} + c_1 \mod 2^{59}$$

$$y_i = k_2 y_{i-1} + c_2 \mod 2^{59},$$

and combined in the manner described below.

The generators may be both mixed, both multiplicative or one mixed and one multiplicative. Later it will be argued that the best combined generator is obtained by using two multiplicative generators. As was noted earlier, the usual practice has been to divide the x_i (or y_i) by 2^{59} to obtain a stream of pseudo-random numbers. Such numbers, while they give every appearance of having been drawn from the population U(0, 1), often do not appear to have been drawn randomly. As the tests applied by MacLaren and Marsaglia demonstrate, there is a short-term correlation between the pseudo-random numbers stretching over spans of at least 10 numbers. The scheme adopted here represents an attempt to destroy the association between contiguous numbers. Suppose, that for x_i and y_i, we label the bits 1 through 59 from least significant to most significant with the 60th (sign) bit always set to zero to indicate a positive number. The 47th through 52nd bits of x_i are interpreted as a binary number (0 - 63) and y_i is subjected to a left circular shift by the amount of this number. The 60th bit of this shifted number is reduced to zero and the shifted y_i and x_i are added, bitwise and modulo 2, to obtain a 59-bit number z_i. This last number is divided by 2^{59} to give the required pseudo-random number. One would like to use the six most significant bits of x_i to determine the shift of y_i, since they are presumably the most nearly random of the 59 bits; on the other hand, since they are the most significant bits and may be the only bits of x_i that have any effect on the final pseudo-random number as used, it seems unreasonable that they should also determine the shift factor for y_i. Choice of the 47th through 52nd bits is thus a more-or-less arbitrary compromise.

There are two 'non-random' features of the scrambling scheme just described which should be noted. The first is that a left circular shift greater than 59 will be interpreted mod 60 so that shifts of 0 through 3 will occur about twice as often as shifts of 4 through 59. The second is that, before shifting, the 60th bit of y_i will be 0 (positive number) but that after shifting this zero will usually occur somewhere in positions 1 through 59. This operation results in the insertion of one non-random

bit into the 59-bit number. The latter problem could easily be removed at the expense of a few more computer instructions but neither of these theoretically undesirable features appears capable of having any really detrimental effect on the randomness of the number z_i.

Before discussing the results obtained for this new generator, a description of the statistical tests employed is given.

4. STATISTICAL TESTS

With one exception the statistical tests employed were identical with those of MacLaren and Marsaglia [2]. All of them were chi-squared tests conducted on samples of 100,000 successive pseudo-random numbers, divided for convenience into 10,000 sets of 10. One complete set of tests was carried out on the same set of 100,000 numbers so that the results are not independent.

Uniformity – The unit interval was divided into 100 equal cells. The first number of each set of 10 was used and the occurrences in each cell counted. Sample size 10,000.

Pairs – Successive pairs of pseudo-random numbers were taken as points in the unit square which was divided into 100 equal cells. Five pairs were used from each set of 10. Sample size: 50,000.

Triples – Successive triples of pseudo-random numbers were taken as points in the unit cube which was divided into 1000 equal cells. Three triples were taken from each set of 10. Sample size: 30,000.

Maximum of n – The maximum of n successive pseudo-random numbers was obtained for n = 2, 3, 4, 5, 10. The n^{th} power of the maximum will also have a uniform distribution and it was tested using the Uniformity Test. One set of n was taken from each set of 10 so that in each case the sample size was 10,000.

Minimum of n – The same tests as for the maximum except that the minimum is used. Note that it is now $(1-\text{Min})^n$ which has a uniform distribution.

Range of n – The range of n successive pseudo-random numbers (i.e., the maximum minus the minimum) was formed for n = 2, 3, 4, 5, 10. The range has a fairly simple distribution and the unit interval, over which the range varies, was divided into 10 equi-probable cells and occurrences in each cell were counted. Sample size was 10,000. This test was used in place of the tests on the sum of n pseudo-random numbers used in [2].

Results of the tests are given in the following manner. The probability of exceeding the computed value of chi-squared on the assumption that the pseudo-random numbers are randomly drawn from the uniform distribution U(0,1) is tabulated as a percentage. Consequently small values of this tabulated probability will be of particular interest to us in suggesting that the assumption is suspect. Values of less than .1% (i.e., a probability of .001) will be marked with an asterisk.

5. RESULTS OF STATISTICAL TESTS

All the evidence in [2] suggests a marked superiority of multiplicative generators over mixed generators. In fact the only non-randomness detected in the output of the former is in the Triples test. It seems reasonable to suppose, then, that the best combined generator would result from combination of two multiplicative generators. This led to our testing the combined generator based on

$$x_i = (2^{31} + 3)\, x_{i-1} \qquad \text{mod } 2^{59}$$

and

$$y_i = (2^{29} + 3)\, y_{i-1} \qquad \text{mod } 2^{59}.$$

At the same time the individual x and y generators were tested in like manner. The individual generators have the maximum cycle length of 2^{57} which will also be the cycle length for the combined generator.

A complete set of tests on the three generators is obtained by testing a sequence of 100,000 numbers from the first generator (with randomly-chosen starting point x_0), a sequence of 100,000 numbers from the second (with randomly-chosen starting point y_0) and a sequence of 100,000 numbers from the combined generator (with starting points x_0 and y_0). Two complete sets of tests are given in the Table together with a third test on the combined generator alone. The x and y generators individually display the marked non-randomness of triples that was discovered by MacLaren and Marsaglia in their tests on multiplicative generators, while the combined generator appears completely satisfactory.

6. CONCLUSION

The pseudo-random uniform number generator based on the combination of two multiplicative generators, as described above, appears to possess satisfactory statistical properties. It maintains the usual advantages of congruential generators - the long cycle and ease of implementation on a digital computer - and does not make any substantial additional demands on the computer memory. In addition, there is the possibility of using one pair of congruential generators to give 2^{57} different cycles (each of length 2^{57}) by combining the congruential generators at each of the 2^{57} positions relative to each other. However, the possible existence of correlations between sequences of numbers from different cycles has not yet been studied. The increased time required for generation of the pseudo-random numbers appears to be a very mild penalty to pay for the improvement in statistical properties. The logical and arithmetic operations involved per pseudo-random number take about 9.8 microseconds on the 6600 as opposed to 5.6 microseconds for the straightforward congruential generator.

7. ACKNOWLEDGMENT

All computer runs made in this study were programmed for the 6600 by Camilla Muriby, who also provided the timing estimates.

TABLE

Test	Combination of $x_i = (2^{31} + 3)\, x_{i-1} \bmod 2^{59}$ and $y_i = (2^{29} + 3)\, y_{i-1} \bmod 2^{59}$			$x_i = (2^{31} + 3)\, x_{i-1} \bmod 2^{59}$		$y_i = (2^{29} + 3)\, y_{i-1} \bmod 2^{59}$	
Uniformity	19	50	99.5	68	73	32	47
Pairs	84	88	31	37	18	7.9	33
Triples	34	98	46	*.0019	.11	*.000043	*.00016
Maximum of 2	22	71	35	77	84	59	86
Maximum of 3	33	32	73	43	96	89	85
Maximum of 4	15	39	66	34	72	16	83
Maximum of 5	37	19	46	32	68	.87	55
Maximum of 10	36	6.2	57	14	75	79	75
Minimum of 2	71	14	96	97	56	84	60
Minimum of 3	82	18	83	55	51	14	99
Minimum of 4	15	61	22	21	43	64	60
Minimum of 5	73	14	67	44	19	74	85
Minimum of 10	54	97	31	24	*.031	64	27
Range of 2	83	77	67	80	72	72	66
Range of 3	76	6.4	80	6.9	32	63	24
Range of 4	5.1	85	37	69	77	25	42
Range of 5	16	9.7	29	17	12	13	3.9
Range of 10	86	99	17	90	33	72	62

REFERENCES

1) T. E. Hull and A. R. Dobell. Random Number Generators. SIAM Review 4 (1962), 230-254.

2) M. D. MacLaren and G. Marsaglia. Uniform Random Number Generators. JACM 12 (1965), 83-89.

CONTROL DATA SALES OFFICES

ALAMAGORDO • ALBUQUERQUE • ATLANTA • AUSTIN, TEXAS • BILLINGS
BOSTON • BOULDER, COLORADO • CAPE CANAVERAL • CHICAGO • CIN-
CINNATI • CLEVELAND • COLORADO SPRINGS • DALLAS • DAYTON • DENVER
DETROIT • DOWNEY, CALIFORNIA • GREENSBORO, NORTH CAROLINA
HONOLULU • HOUSTON • HUNTSVILLE • MIAMI • MONTEREY, CALIFORNIA
INDIANAPOLIS • KANSAS CITY, KANSAS • LOS ANGELES • MADISON,
WISCONSIN • MINNEAPOLIS • NEWARK • NEW ORLEANS • NEW YORK
CITY • OAKLAND • OMAHA • PALO ALTO • PHILADELPHIA • PHOENIX
PITTSBURGH • ROCHESTER, NEW YORK • SACRAMENTO • SALT LAKE
CITY • SAN BERNARDINO • SAN DIEGO • SAN TURCE, PUERTO RICO
SANTA BARBARA • SAN FRANCISCO • SEATTLE • ST. LOUIS • TULSA
WASHINGTON, D. C.

AMSTERDAM • ATHENS • BOMBAY • CANBERRA • DUSSELDORF • FRANK
FURT • HAMBURG • JOHANNESBURG • LONDON • LUCERNE • MELBOURNE
MEXICO CITY • MILAN • MONTREAL • MUNICH • OSLO • OTTAWA • PARIS
TEL AVIV • STOCKHOLM • STUTTGART • SYDNEY • TOKYO (C. ITOH ELEC
TRONIC COMPUTING SERVICE CO., LTD.) • TORONTO • ZURICH

CONTROL DATA CORPORATION

8100 34th AVE. SO., MINNEAPOLIS, MINN. 55440

Pub. No. 6016120(

Litho in U.S.A.